

AD-A065 353

DEFENSE COMMUNICATIONS ENGINEERING CENTER RESTON VA
SOL-370 USER'S GUIDE.(U)
AUG 78 H ULFERS

F/G 9/2

UNCLASSIFIED

DCEC-TN-10-78

SBIE-AD-E100 171

NL

OF
ADA
065353

1



END
DATE
FILMED

4 79
DDC

AD-E 100 171

② LEVEL #
NW

TN 10-78

DEFENSE COMMUNICATIONS ENGINEERING CENTER

AD A0 65353

DDC FILE COPY

TECHNICAL NOTE NO. 10-78

SOL-370
USER'S GUIDE

AUGUST 1978

DDC
RECEIVED
MAR 7 1979
B

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

79 02 16 020

TECHNICAL NOTE NO. 10-78

SOL-370

USER'S GUIDE

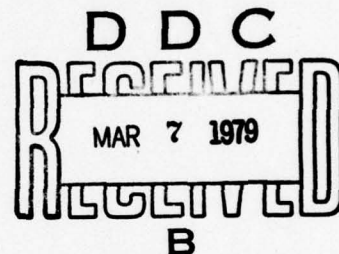
AUG 1978

Prepared by:

Horst Ulfers

Approved for Publication:

Robert E. Lyons
ROBERT E. LYONS
Chief, Computer Systems Division



FOREWORD

The Defense Communications Engineering Center (DCEC) Technical Notes (TN's) are published to inform interested members of the defense community regarding technical activities of the Center, completed and in progress. They are intended to stimulate thinking and encourage information exchange; but they do not represent an approved position or policy of DCEC, and should not be used as authoritative guidance for related planning and/or further action.

Comments or technical inquiries concerning this document are welcome, and should be directed to:

Director
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia, 22090

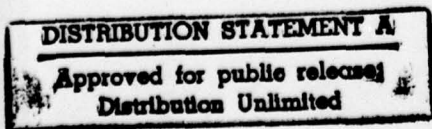


TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. SOL-370 JOB STEPS	2
III. COMPLETION CODES	3
IV. ERROR DUMPS	8
V. TRACING FACILITIES	11
1. Transactions Tracing facilities	11
2. The \$NUMOC Option	13
3. The PL/1 CHECK Option	14
4. the \$SOURCE and \$LIST Options	14
VI. INSTALLATION PROCEDURES	15
1. Systems Datasets	15
2. Initial Tests	16
VII. THE TRANSLATOR	21
VIII. THE COMPILER/LINK-EDITOR	22
IX. THE MODEL EXECUTION	23
X. THE STATISTICS PROGRAM	24
XI. THE PLOT PROGRAM	25
BIBLIOGRAPHY	28

ACCESSION for	
NTIS	Write Section <input checked="" type="checkbox"/>
DDC	Diff Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	Avail. and/or SPECIAL
A	

UNCLASSIFIED

June 1978

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
(14) DCEC-TN-10-78		
4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED	
(6) SOL-370 User's Guide	(9) Technical Note	
7. AUTHOR(s)	6. PERFORMING ORG. REPORT NUMBER	
(10) Horst/Ulfers		
9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	
Defense Communications Engineering Center Computer Systems Division, R800 1860 Wiehle Ave., Reston, VA 22090	N/A (12) 334	
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE	
(Same as 9)	(14) August 1978	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	13. NUMBER OF PAGES	
N/A (18) SBIE (19) AD-E100171	30	
16. DISTRIBUTION STATEMENT (of this Report)	15. SECURITY CLASS. (of this report)	
A. Approved for public release; distribution unlimited.	Unclassified	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
N/A	N/A	
18. SUPPLEMENTARY NOTES		
Review relevance 5 years from submission date.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
SOL-370 Language SOL-370 System System Execution System Installation Error Tracing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
This Technical Note provides information for the user of SOL-370 to install and execute the SOL-370 System, and how to use the error tracing facilities.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407.519

I. INTRODUCTION.

This Technical Note is a companion document to the SOL-370 Language Reference Manual, DCEC Technical Note 11-78 and contains information for the user to install and execute the SOL-370 System, and how to use the error tracing facilities.

All four job steps of the SOL-370 System can be executed individually or in one batch run with four job steps. Some job steps, particularly the Statistics and Plotting Routines, can be run very efficiently in the foreground using the interactive mode. A separate document, TN 16-78, "SOL-370 TSO User's Guide," describes the facilities available for executing the system from a TSO terminal.

II. SOL-370 JOB STEPS

The SOL-370 Simulation System consists of four distinct job steps:

Translate, Compile, Execute, and Analyze.

Figure 1 Illustrates the information flow through the system. The Translator converts the SOL-370 source language deck to PL/1. The PL/1 code generated is compatible with the PL/1-F, PL/1-OPT, and the PL/1 CHECK-OUT Compiler, and is merged with the SOL-370 Library during compilation. The resultant load module should be saved if more executions of the same model are planned. In this case it would be saved in partitioned data set SOL.LOAD. The Execution step of the model will typically write data into two files which are allocated to partitioned data sets SOL.NAMES and SOL.LOG. These two data sets should be saved until all subsequent analysis runs have been performed.

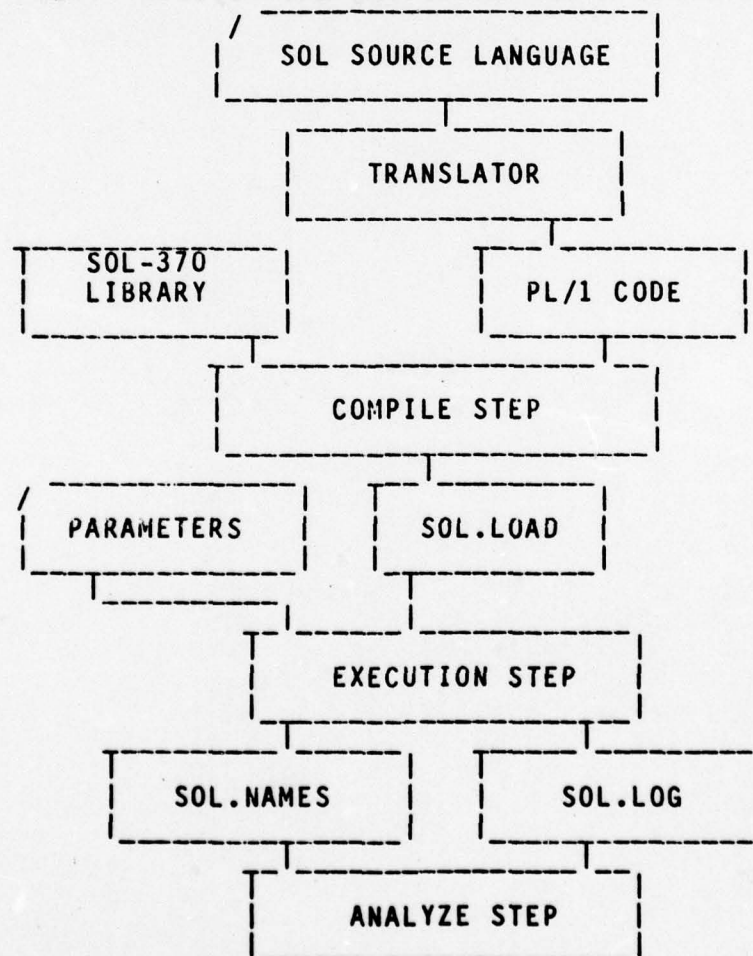


Figure 1. Information Flow

III. COMPLETION CODES

One of the following completion codes is produced at the completion of each simulation run. Any code other than 0000 indicates a run-time error. The error may be caused by a user mistake or systems malfunctioning. In the case of a user abend code, the error is contained in the SOLb source code and the user should carefully check the code. In the case of a system abend code, the user should obtain a listing of the SOL source code, a complete compile listing (compile the model with OPTION(OFFSET)), and the error dump automatically produced when an error has occurred, and submit those listings to SOL Systems Maintenance for analysis. A summary table of completion codes is provided in Table I. Subsequently each completion code is explained in more detail.

TABLE I. SUMMARY TABLE OF SOL COMPLETION CODES.

Compl. Code	Error Type	Related Statement	Error Condition
0000	user	STOP	Successful completion, No error
0001	system	RELEASE	No transaction in facility queue
0010	system	DEMAND	Preempted transaction not found
0011	user	ENTER	Requested amount too large
0100	user	YIELD	Bad trunk name, or not previously demanded, or amount yielded too large
0101	user	RELEASE	Bad facility name, or not previously seized
0110	user	-----	No more active transactions
1000	system	DEMAND	Preempted transaction not found in \$WU and \$INFO arrays
1001	system	LEAVE	Bad store name, or not previously entered, or amount returned too large
1010	user	NEW TRANSACTION TO	Transaction overflow, T in process declarations selected too small
1011	system	\$UPDATE	Transaction seized facility twice
1101	user	SEIZE ENTER DEMAND	Resource overflow, R in process declaration selected too small
1110	user	DISTRIBUTION	Distribution function has too many arguments
1111	system	PART1	PL/1 system error message

System Error Code: 0001

=====

Related SQL statement: RELEASE

Error condition - A transaction releases a facility when other transactions are queued up to seize the same facility. The facility queue does not contain a transaction waiting on this facility, however.

User Action - Save SQL error dump and notify Systems of error condition.

Error flagged in: Procedure RELEAS
locations: 50011240 & 50011290.

System Error Code: 0010

=====

Related SQL statement: DEMAND

Error condition - This error condition is raised when a transaction holding a trunk is preempted and the system cannot find the preempted transaction in the trunk queue.

User Action - Save SQL error dump and notify Systems of condition.

Error flagged in: Procedure \$CONTROL, locations 50002280.

User Error Code: 0011

=====

Related SQL statement: ENTER

Error condition - The amount of storage requested in an enter statement exceeds the declared capacity of this store.

User Action - Check store name and capacity value in the enter statement. If ok, increase the value in the declaration for this store or change the amount requested in the corresponding enter statement.

Error Flagged in: Procedure ENTER, location 50020110.

User Error Code: 0100

=====

RELATED SOL Procedure - YIELD

Error condition - The transaction executing a yield statement has not previously demanded the trunk it is trying to yield now, or the amount to be returned to the trunk is larger than the transaction had taken from the trunk.

User Action - Check for bad trunk name or capacity value in yield statement.

Error flagged in: Procedure YIELD,
Locations - 50033220, 500332600.

User Error Code: 0101

=====

Related SOL statement: RELEASE

Error condition - The transaction executing the release statement has not previously seized the facility it is trying to release now.

User Action - Check for bad facility name used in the release statement.

Error Flagged in: Procedure RELEAS
Location 50010920.

User Error Code: 0110

=====

Related SOL statement: RELEASE

Error condition - The timing routine of the simulator tries to find a transaction in queue to be scheduled for execution. However the queue is empty. This situation may occur, when the model simply runs out of transactions because of slow generation rate, or transactions encounter unrealistic wait conditions.

User Action - No action needed. Insert STOP statement in model source code.

Error Flagged in: Procedure RELEAS
Location 50010920.

System Error Code: 1000

=====

Related SQL statements: DEMAND, YIELD

Error condition - A transaction has been preempted on a trunk. However, an entry for the transaction cannot be found in the \$WU- and \$WUINFO- arrays.

User Action - Save error dump and notify Systems of error condition.

Error Flagged in: Procedure DEMAND,
Locations 50032670, 50032750
Procedure YIELD, Location 50033510
Procedure \$DMENTY,
Locations 50031780, 50031860.

User Error Code: 1001

=====

Related SQL statement: LEAVE

Error condition - The transaction executing the leave statement tries to return capacity to a store it has not previously used, or it tries to return more capacity units than it has taken.

User Action - Check for bad store name or capacity in the leave statement.

Error Flagged in: Procedure LEAVE, Location 50020770.

User Error Code: 1010

=====

Related SQL statement: NEW TRANSACTION TO

Error condition - The execution of the NEW TRANSACTION TO statement creates a new transaction. This error condition is raised when active transactions in the model exceed the number of transactions declared in the process declaration (T=?).

User Action - Increase the value for T in the process declaration.

Error Flagged in: Procedure \$NEWTRN,
Location 50007460, 50007490
Procedure \$LINKIN, LOCATION 50005940

User Error Code: 1011

=====

Related SQL statements: SEIZE, ENTER

Error condition - The same transaction attempts to seize the same facility before releasing it.

User Action - Check model for logical error leading to this error condition.

Error Flagged in: Procedure \$UPDATE,
Location 50006740, 50001100.

User Error Code: 1101

=====

Related SQL statements: SEIZE, ENTER, DEMAND

Error condition - A transaction tries to use more resources (facilities, stores, or trunks) than were specified in the process statement.

User Action - Increase the value for the R parameter in the process statement of the process where the error occurred.

Error Flagged in: Procedure \$UPDATE, LOCATION 50006950
Procedure DEMAND, LOCATION 500328900
Procedure \$DMENTY, LOCATION 50032000.

User Error Code: 1110

=====

Related SQL statement: DISTRIBUTION

Error condition - This error condition is raised when the distribution function contains more than 100 arguments.

User Action - Correct distribution statement.

Error Flagged in: Procedure \$DISTR, LOCATION 50009480.

System Error Code: 1111

=====

Related SQL statement: anywhere

Error condition - This error condition is raised whenever one of the PL/I run-time error conditions is detected. An additional message with the corresponding PL/I abend code is printed.

User Action - Follow IBM instructions to correct error.

IV. ERROR DUMPS

The Error Dump facility has been designed primarily for systems maintenance. For the experienced user, however, it provides a powerful tool in the debugging and verification process of a model. This section explains the output produced in an error dump listing.

An error dump listing is always produced whenever a SOL completion code other than '0000' is produced. Consecutive error dumps may also be programmed for as part of the error tracing facilities as described in chapter V. An error dump listing contains the current value of all global variables, facilities, stores, and trunks, as well as some of the important internal variables and the queue arrays with associated pointers. Figure 2 shows a sample listing which is discussed in the following paragraph.

The number at the end of the first line of the listing corresponds to the SOL statement number where the error occurred .

The headline 'CURRENT VALUES OF GLOBALLY DECLARED ITEMS' is followed by a listing of all facilities, stores, trunks, and global integers with their current values.

The next segment of the listing lists values of all parameters of the transaction that caused the error.

- PROCESS - for the process number,
- IDENTFIR - for the internal transaction identifier,
- RE-ENTRY - for the last reentry label pointer set,
- PRIORITY - for the priority level,
- UPDATPTR - for the last entry in the \$ACTIVE.RES array,
- INTERUPT - for the preemption return label.

The next segment contains the unchanged variables of the transaction at the point when it was reactivated. They are listed in the order in which they were declared.

The following listing of the transactions resource words contains information on all resources the transaction is currently using.

- TYPE - for resource type (facility=0,store=1,trunk=3),
- STRENGTH - for the holding strength applied,
- LINK-INX - for the sequence number of the resource,
- INDEX - for the subscript,if resource is subscripted,
- \$WU PNTR - for the row pointer to the \$WU array,
- AMT - for the amount used on a store or trunk.

The LOCAL VARIABLE listing contains the latest updates of the local variables of the transaction.

The \$FACLNK, \$STRLNK, \$TRKLNK arrays contain the pointers to the \$QUE array for transactions queued up on facilities, stores, and trunks. Each pointer has two values, the first one containing the last queue location entered and the second pointing to the bottom entry of the queue (lowest value). Empty pointers are represented by zero entries.

The GLBLNK ARRAY listing contains the entry points to transaction queues caused by 'WAIT UNTIL' conditions.

The PARAMETER LIST contains the current values of the most important internal simulation parameters:

- \$QMAX - the highest location of \$QUE array used,
- \$NXTQ - pointer to the next location in queue,
- \$QEMTY - pointer to the lowest empty queue location,
- \$TIMLNK - pointer pair to the next transaction to be reactivated (time queue).
- \$TIME - current clock time,
- \$ERRFLG - current value of the error flag.

The \$LNKOAR ARRAY and \$LNKIAR ARRAY listings contain the values of the transactions last linked in or out of the \$QUE array.

The listing of the \$QUE ARRAY consists of the listing of four subarrays: \$QUE1, \$QUE2, \$QUE3, and \$QUE4.

- \$QUE1 - contains the link key,
- \$QUE2 - contains the transaction identifier,
- \$QUE3 - contains special time values if INTERRUPT=WAIT is specified or transactions are preempted on trunks,
- \$QUE4 - contains the pointer to the next queue entry.

The \$WUINFO ARRAY contains pointers to the \$WU ARRAY grouped by holding strength and overflow rows. There are two entries for each column (three entries) in the \$WU ARRAY: the first entry is the pointer to the highest entry in the corresponding \$WU ARRAY row; the second value is the pointer to the overflow row (0 if there is none).

The \$ININF ARRAY contains information on preempted transactions which are awaiting action.

The \$WU ARRAY contains information about transactions that have demanded trunks. There are 15 columns with three entries each: the subscript, the amount, and the transaction identifier. Columns 1 through 8 correspond to the holding strengths 0 through 7. Columns 8 through 15 are overflow rows which are assigned dynamically.

[illegible]

Figure 2. Error Dump Sample Listing

V. TRACING FACILITIES

To verify the logic of a model or to associate error messages with line numbers in the SOL or PL/I source code, the SOL-370 System provides the following tracing facilities.

1. TRANSACTION TRACING FACILITIES

In debugging the model, it may become necessary to trace the flow of a number of transactions through the model. The SOL-370 system provides a special tracing facility to accomplish this. This tracing facility provides for a trace of up to 15 transactions and can be activated at a specific time during simulation. It can also be deactivated at a specific time..

The tracing facility must be invoked at translation time by specifying the following comment card ahead of the program:

```
/*          $TRACE          */
```

Then the SOL source code must be renumbered starting with 10 and subsequently incremented by 10. The following actions will then take place:

a. During translation, the translator will generate a trace call at all labels and at all transfer points to the control module. At these transfer points transactions are usually linked into one of the queue tables and a new transaction is activated. Also, the preprocessor statement is generated to include the librarian module SOLINC(TRACE) and the associated GET statement to read the trace parameters.

b. At compile time the trace module is patched into the program. The trace module must be contained in the SOL library dataset allocated to file SOLINC with the member name of TRACE.

c. During execution of the model, the program will read first the 19 parameter values for the trace routine. These parameters must be the first 19 values in the dataset allocated to file CARD (default allocation is 'SOL.DATA(GOIN)'). The significance of the values read is as follows:

First value:	Simulation clock time at which the Error Dump is activated.
--------------	---

Second value: Simulation clock time at which the Error Dump facility is to be deactivated.

Third value: Simulation clock time at which the trace is to be invoked.

Fourth value: Simulation clock time at which the trace is to be deactivated.

5th -19th value: Transaction identification numbers of the 15 transactions to be traced. There must be 15 values or commas. The transactions are numbered in order of arrival, starting with one transaction each at each process. Process one will be activated first. When a transaction has been cancelled, its number will be reassigned to the next transaction.

The trace printout is placed into the print dataset allocated to print file PRINTER. A sample of the resulting trace follows:

	***** TRANSACTION IDENTIFICATION NO *****									
TIME	1	2	3	4	5	6	7	8	9	10
0	50*									
0	70*									
0	70	90*								
0	70	120*								
0	80*	120								
67	70*	120								
67	70	120	90*							
67	70	120	120*							
101	80*	120	120							
101	80*	120	120							
101	70	120	120	90*						
101	70	120	120	120*						
117	70	130*	120	120						
117	70	150*	120	120						
129	70	*	120	120						
137	70		120	130*						
145	70		120	150*						
156	70	90*	120	150						
171	70	120	120	*						

Each transaction traced is entered into the above table when it executes a statement that puts it into a queue state. This entry is annotated with an asterisk. The leftmost entry on the same row represents the simulation clock time when this entry was made. All entries of transactions in a queue state are repeated without asterisks when the trace of a different transaction is recorded.

For better efficiency, the model should be recompiled without the trace facility when the model has been debugged.

Sometimes, it becomes desirable to produce error dumps independent of the transactions that are traced. This can be accomplished by setting the identifier of the first transaction to be traced (5th value) to zero. An error dump will then be produced each time any transaction is entered into a queue or a label is encountered.

2. THE \$NUMOC OPTION.

The \$NUMOC option of the SOL translator, when enabled at translation time, numbers the lines of the generated PL/1 code in a fashion that allows easy cross-reference between the generated PLI code and the original SOL source code the model was coded in. This feature becomes important when the user wants to relate the IBM run-time error messages, which are annotated with the line number of the PL/1 code, to the corresponding SOL source statement.

To invoke the numbering option, the SOL source code of the model must be preceded by a comment card of the following format:

```
/*          $NUMOC          */
```

The source deck must then be renumbered starting with 10 for the comment card and in subsequent increments of 10. Each line of the generated PL/1 code is then numbered according to the following code:

```
ABBBBBCCCC
```

The first digit 'A' identifies the program module. Only modules 1 or 3 are generated from the SOL model code. The others correspond to the SOL library modules. The second through sixth digits 'BBBBB' correspond directly to the line number of the SOL source code.

3. THE PL/1 CHECK OPTION

The IBM-provided CHECK condition offers a tool for a very detailed trace of variables and labels. To enable this option the user may specify the translator option \$CHECK in the comment card preceding the model source code. The translator will then generate ahead of the PL/1 code the following preprocessor statement:

```
%INCLUDE SOLINC(CHECK);
```

Alternatively, the user may choose to insert this statement into the PL/1 deck directly. A load module compiled with this option will produce a trace of all important SOL variables and labels. Every time a variable assumes a new value, the new value and the associated statement number are printed. These listings may become very long and therefore this feature should only be used in extreme cases. In conjunction with the regular SOL error dump, which is automatically produced when an error occurs, this feature gives the experienced SOL systems programmer a very effective tool to trace a systems error.

4. THE \$SOURCE AND \$LIST OPTIONS

To invoke the \$SOURCE or the \$LIST option the first card of the source deck must be a comment card and must contain the option specified:

```
/*      $SOURCE      $LIST      */
```

When the \$SOURCE option is specified the translator will list each source card when read. The \$LIST option causes the translator to produce a listing of the PL/1 code generated. Both options are helpful when debugging the source code of a model.

VI. INSTALLATION PROCEDURES

1. SYSTEMS DATASETS

The systems tape contains two partitioned datasets :

SYS2.SOLLIB
SYS2.SOLCMDP

The first file "SYS2.SOLLIB" contains all PL/I source library modules as follows:

SOLTRAN - The Translator Module
SOLSTAT - The Statistical Post-Analysis Module
SOLPLOT - The Plotting Routine Module
CHECK - Check Statement for Error Tracing
PART1 - Common Declarations
PART2 - Control Module and Common Procedures
PART3 - Facility Procedures
PART4 - Store Procedures
PART5 - Trunk Procedures
TRACE - Trace Procedure

The second file contains the command procedures, which allow the user to invoke the steps of the SOL-370 System individually or in combination. These procedures will either generate the proper JCL for background submission through the RJE command or invoke the system for foreground execution.

The systems tape can be read with the following JCL code:

```
//GETTAPE JOB (MYID,R820),'MYJOB,U,MYNAME',MSGLEVEL=(1,1),
// NOTIFY=R1111,MSGCLASS=Q,CLASS=A
/*SETUP      JOB REQUIRES TAPE SER NO MYTAPE FOR INPUT      *****
//COPYPDS   EXEC   PGM=IEHMOVE
//SYSPRINT  DD   SYSOUT=(A,U)
//SYSUT1    DD   UNIT=3330,DISP=OLD,VOL=SER=DISK01
//DD1       DD   UNIT=3330,DISP=OLD,VOL=SER=DISK02
//T1 DD DISP=(OLD,KEEP),UNIT=TAPE9,
// DCB=(LRECL=80,BLKSIZE=800,RECFM=FB),VOL=SER=MYTAPE,
// LABEL=(,NL)
//SYSIN DD *
COPY PDS=SYS2.SOLLIB,FROM=TAPE9=(MYTAPE,1),TO=3330=DISK01,      C
FROMDD=T1,CATLG
COPY PDS=SYS2.SOLCMDP,FROM=TAPE9=(MYTAPE,2),TO=3330=DISK01,    C
FROMDD=T1,CATLG
/*
```

Before the SOL-370 System can be operated, three load modules must be compiled and placed into the data set named 'SYS2.SOLLOAD'. The PL/1 source code of the programs to be compiled is contained in the data set 'SYS2.SOLLIB' as members:

SOLTRAN
SOLSTAT
SOLPLOT

The source code of these programs is compatible with all versions of the PL/1-F or PL/1-OPT compilers. For best run-time efficiency, the Optimizing compiler should be used with compile option OPT(2).

2. INITIAL TESTS

To test the installed system, the following model with JCL has been provided. The SOL source code of this model is also stored as member SOLTEST in dataset SYS2.SOLCMOP. Identical results of simulation output and of the analysis step must be obtained for the system to be operational.

```
//R1591TSU JOB (D044,,R830,600,10,250,,),ULFERS,
//          MSGLEVEL=(1,1),NOTIFY=R1591,MSGCLASS=Q
//TRAN EXEC PGM=SOLTRAN,REGION=180K
//TRAN.STEPLIB DD DSN=SYS2.SOLLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)
//TSOOUT DD SYSOUT=(A,U),DCB=(LRECL=131,BLKSIZE=131,RECFM=F)
//OUTF1 DD UNIT=SYSDA,SPACE=(TRK,(5,5)),
//       DCB=(LRECL=80,BLKSIZE=880,RECFM=FB)
//SOLTRAN DD DSN=&&SOLTRAN,DISP=(MOD,PASS),UNIT=SYSDA,
//          SPACE=(400,(60,60)),DCB=(BLKSIZE=400,LRECL=80,RECFM=FB)
//SYSIN DD *
/* TEST OF SOL-370 SYSTEM      $NUMOC $TRACE      */
INTEGER TRKFLG,OLDID,IDFLG,FLAG,CUM(2);
FACILITY TERMINAL(2);
STORE 50 STORAGE(2);
TRUNK 50 BUFFER(2);
PROCESS CONTROL,T=1,R=0;
FLAG1,FLAG2,FLAG3 = 0;
CUM= 0;
TIMER:
FLAG = TIME;
WAIT 1000;
IF TIME < 4000 THEN GO TO TIMER;
STOP;
CANCEL;
END;
```

```

PROCESS LOAD,T=15,R=3;
INTEGER ID,TERM;
ID=0;
OLDID=11;
INTERRUPT=PREEMPT;
START:
ID=ID+1;
NEW TRANSACTION TO LOAD;
WAIT 10;
IF ID<11 THEN GO TO START;
CANCEL;
LOAD:
TERM=MOD(ID,2)+1;
PRIORITY=(10-ID)/2;
SEIZE TERMINAL(TERM),PRIORITY;
WAIT 100;
RELEASE TERMINAL(TERM);
WAIT 5;
IF ID=1 THEN DO;
SEIZE TERMINAL(1),15;
WAIT 5;
RELEASE TERMINAL(1);
END;
IF ID=10 THEN DO;
IF TIME=615 THEN
PUT EDIT('FACILITY QUEUE PRIORITIES PROPERLY PROCESSED')(A) SKIP;
ELSE PUT EDIT('FACILITY QUEUE PRIORITY HANDLING FAULTY')(A) SKIP;
END;
WAIT UNTIL FLAG >= 1000;
IDFLG=ID-OLDID;
OLDID=ID;
IF ID=2 THEN DO;
IF TIME=1000 & IDFLG=1 THEN
PUT EDIT('WAIT-UNTIL QUEUE PRIORITY PROPERLY PROCESSED')(A) SKIP;
ELSE PUT EDIT('WAIT-UNTIL PRIORITY HANDLING FAULTY...')(A) SKIP;
END;
WAIT ID*10;
IF ID=9 THEN DO;
IF TIME=1090 THEN
PUT EDIT('TIME QUE PROPERLY PROCESSED')(A) SKIP;
ELSE PUT EDIT('TIME QUEUE PROCESSING FAULTY.....')(A) SKIP;
END;
ENTER STORAGE(TERM),20;
WAIT 100;
LEAVE STORAGE(TERM),20;
IF ID=5 THEN DO;
IF TIME=1330 & STORAGE(2)=50 THEN
PUT EDIT('STORE QUEUE PRIORITIES PROPERLY PROCESSED')(A) SKIP;
ELSE PUT EDIT('STORE QUEUE PRIORITY HANDLING FAULTY...')(A) SKIP;
END;
WAIT UNTIL FLAG>=2000;
IF ID=3 THEN DO;
ENTER STORAGE(2),10;
END;
WAIT ID*10;
INTERRUPT=INTER;

```

```

TRKLOAD:
DEMAND BUFFER(TERM),20,PRIORITY,PRIORITY;
IF ID=3 THEN DEMAND BUFFER(TERM),10,PRIORITY,PRIORITY;
WAIT 100;
IF ID=3 THEN DO;
    YIELD BUFFER(TERM),30,PRIORITY;
END;
ELSE DO;
    YIELD BUFFER(TERM),20,PRIORITY;
END;
IF ID=1 THEN DO;
    IF TIME=2110 THEN DO;
        PUT EDIT('TRUNK QUEUE PRIORITIES PROPERLY PROCESSED')(A) SKIP;
        WAIT 5;
        DEMAND BUFFER(1),50,7,7;
        WAIT 15;
        YIELD BUFFER(1),50,7,7;
        END;
    ELSE DO;
        PUT EDIT('TRUNK QUEUE PRIORITY HANDLING FAULTY..')(A) SKIP;
    END;
END;
IF ID=11 THEN DO;
    IF TIME=2330 THEN
        PUT EDIT('TRUNK PROCEDURES PROPERLY EXECUTED')(A) SKIP;
    ELSE PUT EDIT('TRUNK PROCESSING FAULTY.....')(A) SKIP;
END;
WAIT UNTIL FLAG >= 3000;
WAIT ID*10,
INTERRUPT = TRKINT;
IF ID > 9 THEN DO;
    IF ID = 10 THEN DO;
        WAIT 10;
        SEIZE TERMINAL(TERM),7;
        DEMAND BUFFER(TERM),50,7,7;
    END;
    IF ID = 11 THEN DO;
        DEMAND BUFFER(TERM),50,7,7;
        SEIZE TERMINAL(TERM),7;
    END;
END;
ELSE DO;
    IF ID = 1 THEN DO;
        SEIZE TERMINAL(1),PRIORITY;
    END;
    DEMAND BUFFER(TERM),10,PRIORITY,PRIORITY;
END;
WAIT 100;
IF ID=10 THEN DO;
    IF CUM(1)=4 & CUM(2)=5 THEN
        PUT EDIT('MIXED TRUNK.FACIL PREEMPTION PROPERLY DONE')(A) SKIP;
    ELSE PUT EDIT('CONCURNT TRUNK/FACIL PREEMPTION FAULTY')(A) SKIP;
END;
CANCEL;

```

```

INTER:
IF ID=4 & TIME=2115 THEN TRKFLG=1;
IF ID=2 & TIME=2115 THEN TRKFLG=TRKFLG+1;
IF TRKFLG=2 & ID=2 THEN
PUT EDIT('TRUNKS PROPERLY PREEMPTED')(A) SKIP;
ELSE IF ID=4 THEN
PUT EDIT('TRUNK PREEMPTION FAULTY.....')(A) SKIP;
WAIT 5;
GO TO TRKLOAD;
PREEMPT:
IF ID=2 & TIME=105 THEN
PUT EDIT('FACILITY PROPERLY PREEMPTED')(A) SKIP;
ELSE IF ID=4 & TIME=110 THEN
PUT EDIT('FACILITY PREEMPTION FAULTY.....')(A) SKIP;
WAIT 5;
GO TO LOAD;
TRKINT:
CUM(TERM)=CUM(TERM)+1;
CANCEL;
END;
END;
/*
// EXEC PL1LFCLG,TIME=5,
// PARM.PL1L='M,NOATR,NS,NS2,EXTDIC,OPT=0,NOSTMT',
// REGION.GO=300K
//PL1L.SYSPRINT DD SYSOUT=(A,U)
//PL1L.SOLINC DD DSN=SYS2.SOLLIB,DISP=SHR
//PL1L.SYSIN DD DSN=SYS2.SOLTRAN,DISP=(OLD,DELETE)
//LKED.SYSLMOD DD DSN=R1591.SOL.LOAD(SOLTEST),
// DCB=(DSORG=PO,LRECL=13030,BLKSIZE=13030,RECFM=U),
// SPACE=(CYL,(1,1,10)),UNIT=3330,DISP=(MOD,CATLG)
//LKED.SYSPRINT DD SYSOUT=(A,U)
//GO.SYSUT1 DD DSN=R1591.SOL.LOAD(SOLTEST),DISP=SHR
//GO.SYSPRINT DD SYSOUT=(A,U)
//GO.SYSIN DD SPACE=(TRK,(1,1)),UNIT=SYSDA
//NAMEFIL DD UNIT=3330,DSN=R1591.SOL.NAMES(SOLTEST),
// DCB=(DSORG=PO,LRECL=63,BLKSIZE=630,RECFM=FB),
// DISP=(MOD,CATLG),SPACE=(TRK,(1,2,5))
//$LOGF DD UNIT=3330,DSN=R1591.SOL.LOG(SOLTEST),
// DCB=(DSORG=PO,LRECL=465,BLKSIZE=465,RECFM=F),
// DISP=(MOD,CATLG),SPACE=(TRK,(10,10,5))
//CARD DD *
10000,10000,10000,10000,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
/*
//PRINTER DD SYSOUT=(A,U)
//DISK DD SYSOUT=(A,U)
//STAT EXEC PGM=SOLSTAT,REGION=200K
//STAT.STEPLIB DD DSN=SYS2.SOLLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=(A,U)

```

```
//SYSIN DD *
NO
0,
NO
NO
10000000,
/*
//LOG DD DSN=R1591.SOL.LOG(SOLTEST),DISP=SHR
//NAME DD DSN=R1591.SOL.NAMES(SOLTEST),DISP=SHR
/*
```

During execution of the model SOLTEST the following output should be produced indicating proper functioning of the SOL-370 Simulation System:

```
FACILITY PROPERLY PREEMPTED
FACILITY QUEUE PRIORITIES PROPERLY PROCESSED
WAIT-UNTIL QUEUE PRIORITIES PROPERLY PROCESSED
TIME QUE PROPERLY PROCESSED
STORE QUEUE PRIORITIES PROPERLY PROCESSED
TRUNK QUEUE PRIORITIES PROPERLY PROCESSED
TRUNKS PROPERLY PREEMPTED
TRUNK PROCEDURES PROPERLY EXECUTED
COMBINED TRUNK.FACIL PREEMPTION PROPERLY DONE
ENDED AT          4000 WITH COMPL.CODE 0000 1 3
```

The statistics step should print the following report if the system is working correctly:

SNAPSHOT TIME IS	10000000						
NAME OF FACILITY	TIME				FRACTION OF TIME IN USE		
TERMINAL (1)	4000				0.2000		
TERMINAL (2)	4000				0.1750		
NAME OF STORE	TIME	CAPCTY	MAX	USD	TOTAL	OCCP	AVG UTL
STORAGE (1)	4000	50		40	10000		0.0500
STORAGE (2)	4000	50		40	23100		0.1155
NAME OF TRUNK	TIME	CAPCTY	MAX	USD	TOTAL	OCCP	AVG UTL
BUFFER (1)	4000	50		50	21550		0.1078
BUFFER (2)	4000	50		50	21000		0.1050

VII. THE TRANSLATOR

The first step is the execution of the SOL translator SOLTRAN. This program converts the model source code to PL/I statements and places the translated code into an intermediate dataset.

The user may specify a number of options which must be contained within a comment:

```
/* $NUMOC, $TRACE, $LIST */
```

\$NUMOC will cause the translator to number the output.

\$TRACE will instruct the translator to generate trace statements.

\$LIST will cause each input line to be printed.

A description for the use of the \$NUMOC and \$TRACE options is contained in chapter V. The \$LIST option simply lists all input card images in addition to the lines which contain errors and would be printed automatically.

The SOLTRAN program requires the following file and dataset allocations:

FILE	DEFAULT DATASET	DATASET DESCRIPTION
SYSIN	SOL.DATA(X)	Model source code in SOL.
SYSPRINT	SYSOUT=(A,U)	Error listing on printer.
TSOOUT	SYSOUT=(A,U)	Input listing on printer.
SOLTRAN	SOL.PLI(X)	PL/I output file on disk.
OUTF1		Temporary work file.

VIII. THE COMPILER/LINK-EDITOR

The compile/link-edit step uses the regular IBM procedures and follows the instructions provided in the corresponding IBM User Guide. The SOL-370 system is compatible with the PL/1-F compiler, the PL/1-Optimizing compiler, and the PL/1-Checkout compiler.

The compile step requires the following file and dataset allocations:

FILE	DEFAULT DATASET	DATASET DESCRIPTION
SYSPRINT	SYSOUT=(A,U)	Compiler listing on printer
SOLINC	SYS2.SOLLIB	SOL-370 library on disk
SYSIN	SOL.PLI(X)	PL/1 input dataset on disk
SYSLIN	SOL.OBJ(X)	Object module, if requested

The Link-Edit step requires the following file and dataset allocations:

FILE	DEFAULT DATASET	DATASET DESCRIPTION
SYSPRINT	SYSOUT=(A,U)	Link-edit listing on printer
SYSLOAD	SOL.LOAD(X)	Load module on disk

IX. MODEL EXECUTION

Once the model has been compiled, it can be executed numerous times with different input parameters. If breakout/restarts are programmed for, the model can be restarted at any one of the checkpoints. The value for the restart point is passed to the program via the PARM parameter of the GO step. However, before a program can be restarted some datasets must be renamed as follows (where X is the name of the model):

In case no restarts of this model have been executed:

```
SOL.RESTART(X) to SOL.RESTART(X#)
SOL.LOG(X) to SOL.LOG(X#)
```

In case one previous restart has been executed and the results are to be saved:

```
SOL.RESTART(X#) to SOL.RESTART(X1)
SOL.LOG(X#) to SOL.LOG(X1)
SOL.RESTART(X) to SOL.RESTART(X#)
SOL.LOG(X) to SOL.LOG(X#)
```

The following file and dataset allocations are required for the Execution step:

FILE	DEFAULT DATASET	DATASET DESCRIPTION
SYSPRINT	SYSOUT=(A,U)	Default printer output
SYSIN		Temporary workfile
NAMEFIL	SOL.NAMES(X)	Simulation names file
\$LOGF	SOL.LOG(X)	Simulation log file
CARD	SOL.DATA(GOIN)	Simulation parameter input
PRINTER	SYSOUT=(A,U)	Printer output
DISK	SYSOUT=(A,U)	Printer Output for traces
\$BRKOUT	SOL.RESTART(X)	Output file at checkpoint
\$BRKFIL	SOL.RESTART(X#)	Input file at restart
\$OLDLOG	SOL.LOG(X#)	Input log file at restart

X. THE STATISTICS PROGRAM

During the execution of the STAT step of the simulation, the Statistics Program SOLSTAT is invoked. This program has been designed as an interactive program for use in the time shared environment. When using this program in the batch environment, proper input data must be prepared equivalent to the action a TSO user would have to take.

The first record read by SOLSTAT specifies whether the tables specified in the source code are to be redimensioned or not. The input should be either YES or NO. If the response is YES then the program prompts the user for new dimensions for all tables. It expects to find for each table three values: the lower limit, the increment, and the upper limit. That is, if the source code had specified three tables and the user wants to redimension only one, he must enter three values for each of the three tables as follows:

```
YES
0,100,1000,
10000,10,10200,
0,10000,100000,
```

The input is in card image format and data entries may start in any column. A character string must not be followed by a comma; numbers may be separated by commas or blanks. If no tables are specified the user should specify NO. The system will then read the next record which specifies the initial number of time units that are to be thrown out for this analysis. A blank record or a comma will invoke the default value zero. This feature is important whenever the user wants to suppress startup effects of a simulation.

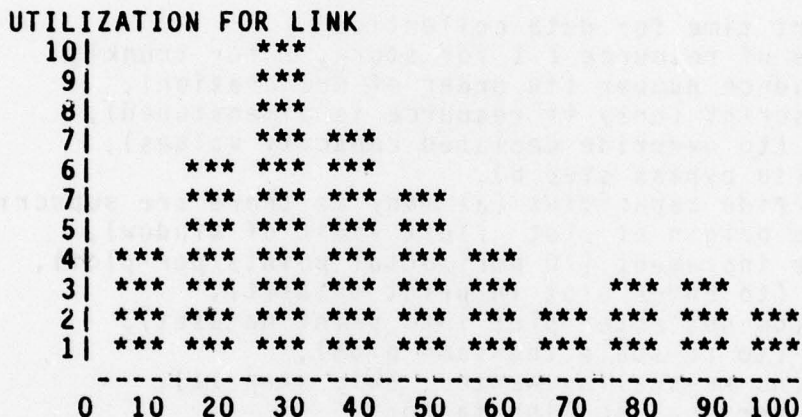
The next record read specifies whether the PLOT routine is to be invoked. This routine is invoked if the user desires a graphical distribution of maximum loads which have occurred on STORES or TRUNKS with dimensions larger than one. The plot function will produce a vertical bar chart, each bar height corresponding to the number of entries for a specific load class. These bar graphs are not too often used but are helpful in illustrating load distributions on Store and Trunk arrays.

If the record read contained a YES then the program will interpret the dual number record on the next card as the type of resource to be plotted (1 for store and 3 for trunk) and the sequence number within each type (numbered in order of appearance in the trunk or store declaration).

The next two records contain scaling factors for the plot. The first record contains the vertical scaling factors (lower limit, increment, upper limit). The second record contains the horizontal scaling factors (lower limit, increment, and upper limit of capacity). The plot specification for a store declared as 100 LINK(50) may look as follows:

```
1,1,
1,1,10,
0,10,100,
```

The resulting bar graph will then look as follows:



The next record contains either a YES or NO and specifies whether more stores or trunks are to be plotted. If the record contains a YES, two more records with scaling specifications for the next plot will be read, etc. If the record contains a NO, the program will interpret the next record as a specification for snapshot statistics. If the record contains a NO, only one statistics listing for the time the simulation terminated is produced. If the record contains a YES, then subsequent records are read, each containing the specific value of the simulation time for which the statistics snapshot is to be produced for. The program will stop when it either finds a specification for a snapshot outside the simulation run or when there are no more records.

The SOLSTAT program requires the following file and dataset allocations:

FILE	DEFAULT DATASET	DATASET DESCRIPTION
LOG	SOL.LOG(X)	Simulation log file
NAMME	SOL.NAMES(X)	Simulation names file
SYSIN	SOL.DATA(STATIN)	Statistics parameter input
SYSPRINT	SYSOUT=(A,U)	Printer output file

XI. THE PLOT PROGRAM

The PLOT program is the second of the program products available for post-simulation analysis. This program provides the capability to plot loads on stores or trunks as a function of simulation time. Besides the regular output it also generates a special print dataset, which can be used to print formatted plots on special paper. An example is given in Figure 4. The program was developed for use in the TSO environment and is highly interactive. However it can also be run in the batch mode either using the built-in defaults (empty input records) or by preparing the input data according to the protocol of the interactive conversational mode. To do this the following protocol is provided:

1. Start time for data collection,
2. Type of resource (1 for store, 2 for trunk),
3. Sequence number (in order of declaration),
4. Subscript (only if resource is dimensioned),
5. YES (to override declared capacity values),
NO (to bypass step 6),
6. Override capacities (as many as there are subscripts),
7. Time origin of plot (left limit of window),
8. Time increment (70 horizontal points/per plot),
9. YES (to enter plot in print dataset),
NO (do not enter plot into print dataset),
10. YES (to re-scale the same plot),
NO (no re-scaling wanted, skip step 11),
11. New vertical scaling factor,
Return to 7,
12. YES (to plot other resources, go to step 2),
NO (to terminate processing),
13. YES (if it was last possible resource and PLOT is
to restart, go to step 1),
14. NO (to terminate processing),

The SOLPLOT program requires the following file and dataset allocations

FILE	DEFAULT DATASET	DATASET DESCRIPTION
LOG	SOL.LOG(X)	Simulation log file
NAMME	SOL.NAMES(X)	Simulation names file
CRD	SOL.DATA(PLOTIN)	Plot parameter file
OUT1	X.DATA	Plot print file
SYSPRINT	X.OUTLIST	Dummy print file
OUT	SYSOUT=(A,U)	Printer output file

The following JCL listing is an example of executing the SOLPLOT program in the batch environment. Figure 3 has been generated by this run.

```

//R1591PO JOB (D044,R830,440,15,180,,),ULFERS,
//          MSGLEVEL=(1,1),NOTIFY=R1591,MSGCLASS=Q
//PLOT EXEC PGM=TSOPLOT,REGION=180K,TIME=4,
//          PARM=' '
//STEPLIB DD DSN=UR1591.LOAD,DISP=(SHR,KEEP)
//SYSPRINT DD SYSOUT=(Y,U)
//CRD DD *
0
3
1
YES
50,50
2000
10
NO
YES
5
2000
20
YES
NO
END
//LOG DD DSN=R1591.SOL.LOG(SOLTEST),DISP=SHR
//NAME DD DSN=R1591.SOL.NAMES(SOLTEST),DISP=SHR
//OUT DD SYSOUT=(A,U)
//OUT1 DD DSN=R1591.SOLTEST.DATA,UNIT=SYSTS,SPACE=(1680,(100,600)),
//       DCB=(LRECL=80,BLKSIZE=1680,RECFM=FBA),DISP=(MOD,CATLG)

```

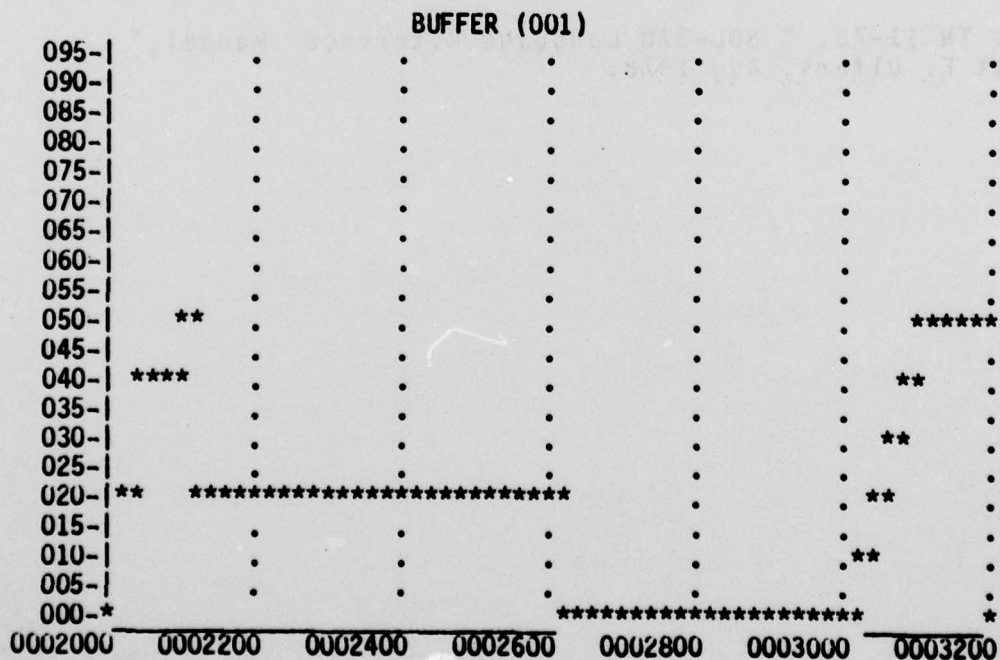


Figure 3. Example of PLOT Routine Output

BIBLIOGRAPHY

1. D. E. Knuth and J. L. McNeley, "A Formal Definition of SOL," IEEE Transactions on Electronic Computers, EC-13 No. 5 (Aug 1964) pp 409-414.
2. D. E. Knuth and J. L. McNeley, "SOL - A Symbolic Language for General Purpose Systems Simulation," IEEE Transactions on Electronic Computers, IC-13, No. 5 (Aug 1964) pp 401-408.
3. R&D Technical Report ECOM-3085 (AD-850159L), "MALLARD Traffic Simulation, Results and Analysis, Final Report," James A. Armstrong and Horst E. Ulfers, Feb 1969.
4. J. Armstrong, H. Ulfers, D. Miller, H. Page, "SOLPASS - A Simulation Oriented Language Programming and Simulation System," Proceedings of the Third Conference on Applications of Simulation, Dec 1969.
5. R&D Technical Report ECOM-0043-F, "SOL Compiler Design," H.C. Page, D.J. Miller (Patterson & Smith Inc), Feb 1968.
6. Horst E. Ulfers, "PACKNET - A Packet Switch Network Simulator," Proceedings of the 1975 ICC, June 1975.
7. C. G. Guffee and H. E. Ulfers, "SOL-370," Proceedings of the 1975 Summer Computer Simulation Conference, July 1975, pp 1-11.
8. DCEC TN 11-78, "SOL-370 Language Reference Manual," Horst E. Ulfers, Aug 1978.

DISTRIBUTION LIST

R100 - 2	R200 - 1
R102/R103/R103R - 1	R300 - 1
R102M - 1	R400 - 1
R102T - 9 (8 for stock)	R500 - 1
R104 - 1	R700 - 1
R110 - 1	R800 - 1
R123A - 1 (Library)	NCS-TS - 1
R124A - 1 (for Archives)	

205 - 20

DCA-EUR - 1 (Defense Communications Agency European Area
ATTN: Technical Director
APO New York 09131)

DCA-PAC - 1 (Defense Communications Agency Pacific Area
ATTN: Technical Director
Wheeler AFB, HI 96854)

USDCFO - 1 (Unclassified/Unlimited Distribution)
(Chief, USDCFO/US NATO
APO New York, 09667)

SPECIAL:

R830 - 100